Cooperating Expert Systems for the Next Generation of Real-time Monitoring Applications

U. M. Schwuttke, J. R. Veregge, and A. G. Quan
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA911 09
Tel: 818-354-1414 Fax: 818-393-6004
urns @puente.j pl .nasa. gov

ABSTRACT

A distributed monitoring and diagnosis system has been developed and successfully applied to real--time monitoring of interplanetary spacecraft at NASA's Jet Propulsion Laboratory. This system uses a combination of conventional processing and artificial intelligence. Knowledge-based diagnosis modules are embedded within an automated monitoring system that detects on-board spacecraft anomalies. The diagnostic modules are specialized to respond to anomalies in a single domain of expertise and to cooperate with one another when necessary to solve complex problems that extend beyond an individual domain. Details of the distributed architecture, real-time diagnosis, and system performance are described in the paper. A brief summary of lessons learned in transferring research prototypes into operational environments is also reported.

Key Words: Automation, distributed systems, expert systems, monitoring and diagnosis, real-tiJnc.

INTRODUCTION

A combination of' practical and innovative computer science has been applied to the MARVEL system [Schwuttke et al. 1992] for automated monitoring and diagnosis of spacecraft telemetry. This system has been shown to achieve robust and coherent behavior for complex, real-time diagnostic modules embedded in a conventional (algorithmic) monitoring system.

The system architecture has been designed to facilitate concurrent and cooperative processing by multiple diagnostic expert systems in a hierarchical organization. The expert systems adhere to concepts of data-driven reasoning, constrained but complete nonoverlapping domains, metaknowledge of global consequences of anomalous data, hierarchical reporting of problems that extend beyond a single domain, and shared responsibility for problems that overlap domains.

These features combine to enable efficient diagnosis of complex system failures in real-time environments with high data volumes and moderate failure rates, as indicated by detailed performance

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology under a coat ract with the National Aeronautics and Space Administrate ion. '1 'he authors wish to acknowledge support from JPL's Voyager and Galileo Projects, Multimission Ground Support Office and Director's Discretionary Fund.

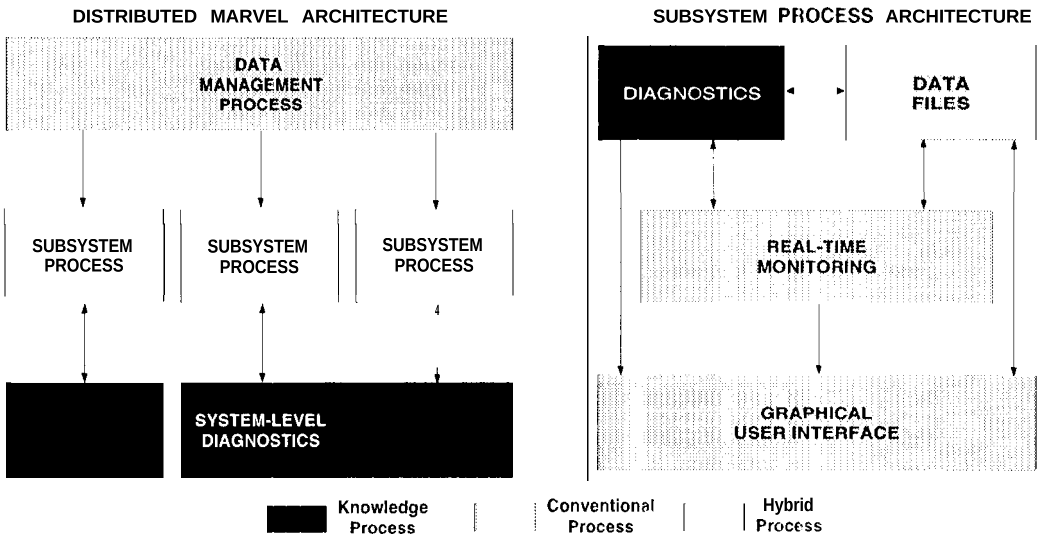

Figure 1. The distributed architecture on the left can currently be configured to run on one to four UNIX workstations, with the most common operational configuration involving two workstations (for compatibility with analyst responsibilities). The hybrid subsystem processes on the left are composed of conventional and knowledge processes, as shown in the figure on the right.

measurements from two different applications of the system. One of these applications has been in continuous operational use since it was first deployed in" 1989 for the Voyager spacecraft encounter with Neptune. This application remained under incremental development for a period of three years subsequent to the original delivery and has been under routine maintenance since 1991. The current application for the Galileo mission is a second generation system that has been on-line for only one year and is still under active development. The second generation system builds on experience gained with this technology to achieve an order of magnitude increase in performance.

COOPERATING EXPERT SYSTEMS EMBEDDED IN A DISTRIBUTED ARCHITECTURE

Recently, the need for mechanisms of cooperation that are sufficiently robust for real-world monitoring applications has become a research driver. Systems such as GRATE* [Jennings and Mamdani, 1992] contribute toward a clearer and more easily implementable interaction of agents during collaborative problem solving. GRATE* addresses a problem domain in which events occur unpredictably and decisions may be based on incomplete or imprecise data. Toward this end, the notion of joint responsibility is proposed as an alternative to the more conventional notion of agents acting in self-interest, The potential for large communication overhead is a possible disadvantage of the GRATE* system, particularly for applications with time critical analysis.

The protocol and architecture described in this paper builds on the notion of joint responsibility and uses modular problem decomposition and data-driven reasoning in order to minimize communication between agents. The various modules in the distributed architecture of Figure 1 are allocated among a configuration of UNIX workstations. Interprocess communication is based on a central message routing scheme. The data management module receives data from a source (in the case of our current application, the data is spacecraft telemetry received from JPL's ground data system) and allocates it to the appropriate subsystem monitor based on identification of data type. (Our system is partitioned according to the structure of the spacecraft, with one subsystem monitor for every spacecraft subsystem. Spacecraft subsystems include command and data, attitude and articulation control, propulsion, telecommunications, thermal, and power. A mapping between partitioning in the monitoring system and the natural partitioning of the system being monitored is desirable for real-time diagnostic architectures.) Each of the subsystem monitors provides algorithmic functions such as validation of telemetry, detection of anomalies, trend analysis and automatic reporting. These functions, while not in themselves of interest in AI or computer science research, are vital components of a real-world diagnostic system. They are implemented here in conventional C code for performance reasons. In addition, each subsystem process can provide diagnosis of failures based on anomalous data and recommendation of corrective actions. The latter two functions are provided by knowledge-based modules that are embedded within each of the individual subsystem monitors. The remaining modules include the graphical user interface and display processes for each of the subsystem monitors, and the system-level diagnostic agent for handling failures that manifest themselves across multiple subsystems (and therefore cannot be completely analyzed by any one subsystem alone). Detailed reasoning examples from the actual application are presented elsewhere [Schwuttke and Quan 1993].

CHARACTERISTICS OF THE EXPERT SYSTEMS

The Expert Systems are Embedded

Rule-based diagnostic modules are embedded in efficient algorithmic code. The algorithmic code performs all functions that do not explicitly require reasoning capability, so that the use of the less efficient reasoning modules is limited to those functions for which it is essential.

Diagnosis is Data-driven

Forward-chaining demons are used 10 represent domain knowledge. Reasoning is activated by the appearance of data that requires diagnosis. The initial determination that diagnosis is required is made by algorithmic monitoring code, which detects potential anomalies algorithmically and passes the anomalous data to an appropriate diagnostician. in the absence of anomalous data within its domain, a diagnostic system is idle.

The Domain of Individual Experts is Constrained

An agent is responsible for a small, clearly partitionable domain of expertise. Partitioning is governed by the natural decomposition of the system being diagnosed. 'l'his helps overcome disadvantages associated with rule-based systems for which, typically, implementation can be intractable, execution is nondeterministic and relatively slow, and verification can be difficult. Small, modular knowledge-bases enable developers to handle more easily definable subproblems. Smaller knowledge bases execute more efficiently, because less time is spent in search. Finally, smaller knowledge-bases are easier to verify.

The Domain of the Individual Diagnostic Modules is Nonoverlapping

A particular domain of expertise and the associated rules for performing diagnosis are assigned only to one diagnostic module in order to avoid redundant reasoning.

Diagnostic modules carry individual responsibility for problems entirely within their domain. Each diagnostician has sufficient knowledge to be fully accountable for diagnoses within its area and has no knowledge of other domains. This requires that accountability for locally detectable failures must be local,

Failure Domains May Not Map Directly to Agent Domains

Diagnosis requires more than one agent when the symptoms manifest themselves in more than one domain,

Metaknowledge Enables Agents to Instigate Cooperation for Diagnosis Beyond Their Domain

Agents have metaknowledge to identify symptoms of failures that could possibly extend beyond

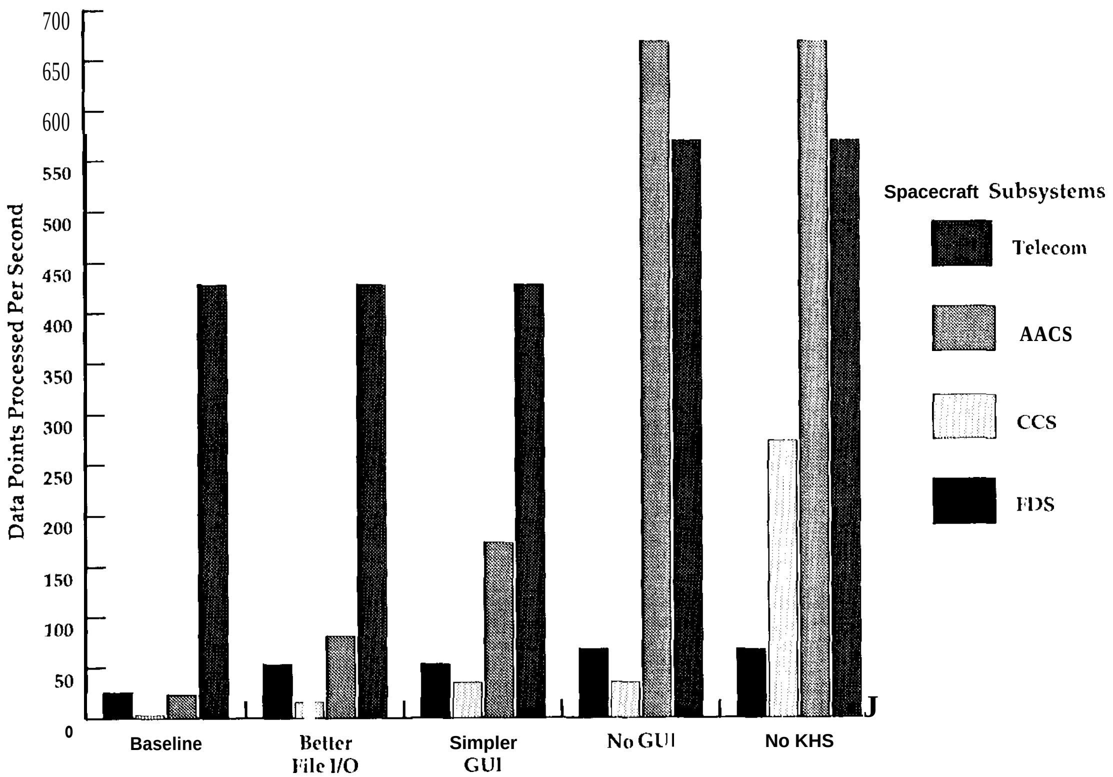

Figure 2. Performance results for each of the subsystem modules.

their domain. Metaknowledge is contained in a set of rules in each knowledge-base, and is associated with the occurrence of events whose analysis may require the cooperation of other agents.

Agents Report All Problems That Extend Beyond Their Domain

Metaknowledge enables an agent to determine which symptoms from its domain may portend problems beyond its domain. The metaknowledge also includes the specific agent(s) to which the information should be forwarded.

A Hierarchy of Agents Provides Coordination

An expert forwards all known information pertaining to failures beyond its domain to another agent at the next higher level in the hierarchy. The underlying approach on forwarded messages is conservative; it is up to the agent receiving the information to determine whether a fault requiring a diagnostic message and an alarm has occurred or whether the anomalous data has some other explanation. This agent may also receive messages from other lower-level agents. Experts at the. higher level are implemented according to the same principles as lower-level experts; thus reasoning at the higher levels of the hierarchy is also data driven. The agents at the higher level are activated by message.s from lower-level agents, just as the lowest level agents were activated by messages of symptoms detected by algorithmic code. Messages are directed with metaknowledge to the relevant agent(s) in order to complete the final analysis of the anomalous data and provide diagnosis of any associated failures.

Agents Share Responsibility for Diagnosis of Problems That Overlap Domains

Joint responsibility exists in that the lower-level agents are responsible for reporting appropriate symptoms upward in the hierarchy and the higher-level agent(s) are responsible for correctly determining whether failures have occurred and providing appropriate diagnosis. This differs from the "self interest" model of communication [Durfee 1988] and is similar to the joint responsibility model [Jennings and Mamdani, 1992] in which agents must temper their self-interest with consideration to a group. These models have parallels in social organizations, with the first being more representative of an unstructured society and the second paralleling the actions of individuals who are dedicated (perhaps for reasons of self-interest) to fulfilling a successful role in a structured organization such as a business or a corporation. in the latter case, independent agents work together with appropriate (and hierarchical) division of responsibility towards fulfilling a common goal. Real-world applications can be sufficiently complex that only this second type of organization may enable timely, robust, and coherent behavior.

EXPERIMENTAL RESULTS

The distributed architecture described in this paper has been applied to two generations of real-time monitoring systems. The Galileo system, currently under development, does not yet include modules for diagnosis. The Voyager system, completed in 1991, contains four diagnostic expert systems (developed using a commercial shell) in a two-level hierarchy.

Conventional monitoring modules for four of the spacecraft subsystems were completed: the flight data subsystem, the computer command subsystem, the attitude and articulation control subsystem, and the telecom subsystem. Three of our expert systems are embedded in conventional modules that perform data access/manipulation and monitoring in addition to providing graphical user interfaces and other subsystem specific automation. The system-level diagnostician is not embedded within another module. As a result, it cannot easily be compared to the other expert systems in a discussion of real-time performance and it will not be further discussed here.

The remaining expert systems have the following characteristics. The computer command subsystem (CCS) expert contains on the order of 150 rules, focuses on a relatively broad domain analysis, and is invoked very frequently (for almost every parameter). The attitude and articulation control subsystem (AACS) expert contains approximately 100 rules, and focuses on a more narrow domain of analysis. It is invoked infrequently. The telecom expert system contains on the order of twenty-five rules and is invoked continuously (for every parameter). The flight data subsystem (FDS) module does not contain an expert system.

Experimental evaluation on a network of workstations (Sun Microsystem Spare LXs running Solaris 2.2) involved a series of tests to determine the maximum number of data parameters that could be processed per module per second (a subsystem module includes both the conventional and knowledge-based components as shown in Figure 1). The primary purpose of this evaluation was to learn about the performance of the expert systems and apply our insights to future expert system implementation on the Galileo application. This evaluation was not motivated by a need to improve the performance of the Voyager system, as current data rates are considerably slower than during the planetary encounters and are easily handled by the existing software configuration.

The results are shown in Figure 2. The baseline performance was below expectation, with FDS, CCS, AACS and Telecom processing 26,3,24, and 428 parameters per second respectively, for a total of 481 parameters per second processed by the entire system. Performance profiling revealed that file 1/0 and the graphical user interfaces (GUIs) were primary performance bottlenecks.

With regard to these bottlenecks, the four modules can be categorized as follows. FDS and CDS have moderately complex GUIs, and perform significant file 1/0. AACS has the most complex GUI and performs very little file 1/0, because the input files read by this subsystem are sufficiently small that they are read entirely into memory upon system initialization. Telecom has a simple GUI and performs no file 1/0.

Optimizing file 1/0 where possible improved performance to 53, 16, 81, and 428 parameters per second. (This is the only improvement discussed in this section that was carried forward to the operational system.) Simplifying the graphical user interface by eliminating real-time scrolling windows (known to be computationally inefficient in MOTIF user interfaces; considered desirable by end-users and thus included in the FDS, CCS, and AACS modules of the operational system) further improved performance to 53, 35, 172, and 428 parameters per second. Eliminating the graphical user interface entirely resulted in performance increases to 67, 35, 646, and 570 parameters per second. Finally, eliminating the expert systems yielded performance of 67, 273, 668, and 570 parameters per second.

These results made it possible to gain a number of new insights with regard to our system. The biggest surprise was the high performance of the telecom module. The combination of the small knowledge base and the simple user interface enables processing of 428 parameters per second. Elimination of both the GUI and the expert system only results in a further performance improvement on the order of 25 percent, indicating that no substantial] penalty is associated with the significant enhancement to functionality provided by these two components of the module. The next generation system will benefit from this result, in that frequently performed analysis that requires the use of an expert system will be implemented with a number of small, cooperating modules rather than one larger module. Further performance improvement could likely be gained with a more efficient expert system shell. This will be investigated although we do not current] y expect more than a several-folcl improvement.

The AACS expert system is larger by a factor of four, and slower by an order of magnitude. This can be explained by both a larger search space and greater depth in each search. Performance could likely be improved with a faster reasoning shell and by modularization of the knowledge base. However, the diagnostic component of this module is invoked sufficiently rarely (less than once per hour) that this is not an important bottleneck as there would be insufficient opportunity to benefit from this improvement. In the case of this type of module, it would be preferable to simplify the GUI, which continues to impose considerable resource overhead.

The CCS expert system is large and is invoked regularly as par-t of ongoing trend analysis in that subsystem module. Elimination of the expert system results in an additional order of magnitude increase in performance, providing further i ndicat ion that a large knowledge base may be inappropri ate for frequently invoked real-time diagnosis. The CCS knowledge base is characterized by breadth rather than depth. As a result, it would both beneficial (and straightforward) to reduce it to three or more component modules without imposing significant overhead from resulting interprocess communication. (If this were implemented, the CCS module would still be 1/0 bound, as it reads from a number of very large files.)

As a result of these insights, the Galileo implementation takes a more efficient approach to file 1/0. It also tends to be more efficient in its graphical user interface, in that it does not include some of the higher-overhead user interface widgets. Such changes impact functionality, requiring a certain amount of negotiation with end-users (who are typically willing to compromise in favor of performance). In addition, the Galileo system makes greater use of the distributed architecture with more than one module per subsystem. With these changes we are currently able to process a three-fold increase in telemetry parameters in the baseline configuration. In the future, the addition of small, modular expert systems for diagnosis is planned. These will be implemented to have the minimum impact on performance.

OTHER LESSONS LEARNED IN THE TRANSITION BETWEEN RESEARCH AND OPERATIONS

The development of MARVEL has involved a constant balance between user needs, research goals, and (less conveniently) retrofit to an existing oper ations system that was never intended for automat ion. Under such circumstances, the temptation to put research goals ahead of al l other considerations is common; however, these goals must be balanced against user needs in order to maintain the customer support that is needed to assure long-term survival. In many cases, sufficient communication with customers can actually help focus research on real needs. The following lessons have been valuable in making a successful transition to operations.

Existing Tools Enhance Development Progress

Reasonably priced commercial tools and public domain tools were used in MARVEL with great success, for expert system development and for conventional functions such as graphical user interfaces, trend plotting, and network communication. This turned out to be an advantage from both the implementation and maintenance perspectives, allowing cost-effective software development to concentrate on unique task needs for which there were no tools. Recently, some in-house software has even begun to emerge that could be effectively reused for some of these unique needs.

Knowledge-based Methods Should Be Used Sparingly in Real-time Systems

For diagnosis functions expert systems provide better implementational paradigms than more efficient convent ional approaches. However, expert systems usual l y emplo y interpreters to perform in ferencing on the knowledge base rather than compiling the knowledge base into native code. This tends to compromise performance and can pose difficulties in applications where the fastest possible response time is a critical factor in meeting real-time constraints [Bahr and Barachini 1990].

MARVEL achieves adequate response time by placing as much of the computing burden as possible into conventional algorithmic functions written in C. For example, C processes handle the initial tasks of allocating telemetry to a monitoring module and detecting anomalies. After preliminary tests are done and a probability of anomaly occurrence has been established, the subsystem monitor invokes knowledge-based processing for diagnosis of the anomaly and for recommendation of corrective action. This technique contributes to an overall response time that is sufficient for real-time monitoring.

There is More Than One Way to Benefit From a Diagnostic System

initial emphasis using MARVEL for productivity enhancement temporarily curtailed the development of diagnostic expert systems; it was perceived that diagnostic systems did not improve efficiency of operat ions. This perception stemmed from t wo observations: First, anomal y anal ysis was only required in the presence of spacecraft anomalies. Second, these did not occur with sufficient frequent y to warrant an automated approach, particular] y since human confirmat ion of the expert s ystem anal ysis would still be required.

1 lowever, mandated workforce reductions subsequent to the Neptune encounter caused renewed interest in expert systems. Now the goal is no longer workforce reduction, but the preservation of mission expertise. Many current analysts arc new to the mission and, for the most part, do not have the experience of the previous staff. The ncw personnel will have fewer opportunities to gain such experience: although the Voyager interstellar mission is scheduled to continue until approximately 2018, spacecraft activity is at a low level. As a result, there arc far fewer opportunities for mission operations personnel to learn about the spacecraft and its operation (ban during the prime mission. There is concern that analysts with the experience to handle future anomalies will be less readily available, or that they will have retired. As a result, the expert systems arc being expanded to provide information that is based on the expertise of former analysts. However, this is much more difficult than it would have been several years ago, as many of the experts arc no longer available.

Successful Automation Emphasizes Depth Over Breadth

Emphasis on depth over breadth in automation applies equally to conventional components of a system and to expert systems. Attempt to establish viability by simultaneously demonstrating functionality in a the many diverse areas relevant to a large application can result in the inability to achieve focus in most of these areas. In MARVEL, depth has been more difficult to achieve in the expert systems than in the conventional components of the system. This has resulted from two factors. The first of these was the need to strive for a system that would enable workforce reductions. The second reason is that it was more difficult to elicit user requirements and domain knowledge for expert systems than for conventional functions, perhaps because the analysts were better able to express knowledge and communicate information regarding more conventional or algorithmically oriented tasks.

CONCLUSIONS

The MARVEL distributed architecture demonstrates the successful implementation of multiple cooperating agents in a complex rca]-time diagnostic system. Wc have designed an architecture that facilitates concurrent and cooperative processing by multiple agents in a hierarchical organization. These agents adhere to the concepts of dat a-driven embedded diagnosis, constrai ned but complete nonoverlapping domains, metaknowledge of global consequences of anomalous data, hierarchical reporting of problems that extend beyond an agent's domain, and shared responsibility for problems that overlap domains.

The MARVEL architecture is simple and well suited for real-time telemetry analysis. Conventional processing is used wherever possible in order to facilitate performance. The knowledge-based agents are embedded within the algorithmic code, and are invoked only when necessary for diagnostic reasoning. Distribution of telemetry monitoring and diagnostic processes across workstations provides significant improvement in performance. These qualities allow for efficient real-time diagnosis of anomal ics occurring in a complex application.

Maximum modularization of frequently invoked reasoning modules will enable significant performance improvements in the next generation system.

REFERENCES

Bahr, E.; and Barachini, F. 1990. "Parallel PAMELA on PRE.'' in Parallel Processing of Engineering Applications, ed, R. A. Adcy, 209-219. New York: Springer-Verla.g.

Cohen, P. R.; Hart, D. M.; and Howe, A. E. 1990. "Addressing Re:il-time. Constraints in the Design of Autonomous Agents.' ' COINS Technical Report 90-06. Universit y of Massachusetts at Amherst.

Durfee, 13.11. 1988. "Cooperation through Communication in a Distributed Problem Solving Network. " In Distributed Artificial Intelligence, Vol. 2. Pitman Publishing, 1988.

Jennings, N. R.; and Mamdani, E. H. 1992. ''Using Joint Responsibility y to Coordinate Collaborative Problem Solving in Dynamic Environments. " In Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, California. 269-275.

Hayes-Roth, B. 1990. "Architectural Foundations for Real-time Performance." Artificial Intelligence Journal, 26:251-232.

Horvitz, E. J.; Cooper, G. F.; and Heckerman, D. E. 1989. "Reflection and Action Under Scarce Resources: Theoretical Principles and Empirical Study. " In Proceedings of the Eleventh international Joint Conference on Artificial Intelligence, 1121-1127.

Schwuttke, U. M.; Quan, A. G.; Angelino, R.; Childs, C. L.; Veregge, J. R.; Yeung, R.; and Rivera, M. 13.1992. ''MARVEL: A Distributed Real-time Monitoring and Analysis Application. " In Innovative Applications of Artificial Intelligence 4, MIT Press.

Schwuttke, U. M.; and Quan, A. G. 1993. "Enhancing Performance of Cooperating Agents in Real-time Diagnostic Systems.'' In Proceedings of the Thirteenth international Joint Conference on Artificial Intelligence, Chambery, France, 332-337.